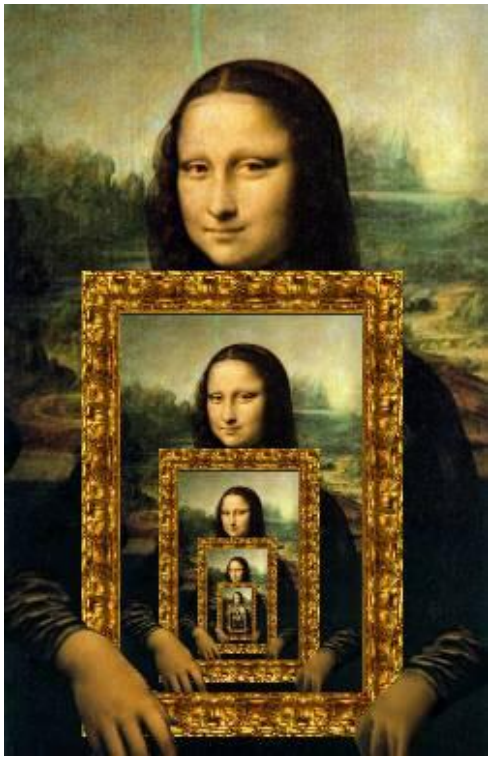


Recursividade

Prof. Demétrios Coutinho

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Campus Pau dos Ferros
Disciplina de Organização de Algoritmos
Demetrios.coutinho@ifrn.edu.br

Definição de Recursão

É o nome que se dá quando uma função chama a si própria.

- Existe a recursão direta – quando uma função chama a si mesma diretamente.
- E a recursão indireta – quando uma função chama outra, e esta, por sua vez chama a primeira.

Uma função pode ser implementada de forma iterativa ou recursiva.

- O código de uma função recursiva é mais simples e clara.
- Implementações iterativas tendem a ser mais eficientes (performance) que as recursivas.

Definição de Recursão

Sempre que há uma chamada de função (recursiva ou não) os parâmetros e as variáveis locais são empilhadas na pilha de execução.

No caso da função recursiva, para cada chamada é criado um ambiente local próprio. (As variáveis locais de chamadas recursivas são independentes entre si, como se fossem provenientes de funções diferentes).

Definição de Recursão

Uma função pode chamar a si própria por um número limitado de vezes.

Esse limite é dado pelo tamanho da pilha. Se o valor correspondente ao tamanho máximo da pilha for atingido, haverá um estouro da pilha ou **Stack Overflow**.

Cada vez que uma função é chamada de forma recursiva, são alojados e armazenados uma cópia dos seus parâmetros, de modo a não perder os valores dos parâmetros das chamadas anteriores.

Ao final da execução,os dados são desempilhados e a execução volta ao subprograma que chamou a função.

Elementos da Recursividade

Ponto de Parada ou **Condição de Parada**: é o ponto onde a função será encerrada.

Regra Geral: é o método que reduz a resolução do problema através da invocação recursiva de casos menores, que por sua vez são resolvidos pela resolução de casos ainda menores pela própria função, assim sucessivamente até atingir o "ponto de parada" que finaliza a função.

Elementos da Recursividade

Exemplo Fatorial

A recursividade é uma estratégia que pode ser utilizada sempre que o cálculo de uma função para o valor n, pode ser descrita a partir do cálculo desta mesma função para o termo anterior (n-1).

Fatorial:

4! = ?

4! = 4*3!

4! = 4*3*2!

4! = 4*3*2*1!

4! = 4*3*2*1*0!

4! = 4*3*2*1*1

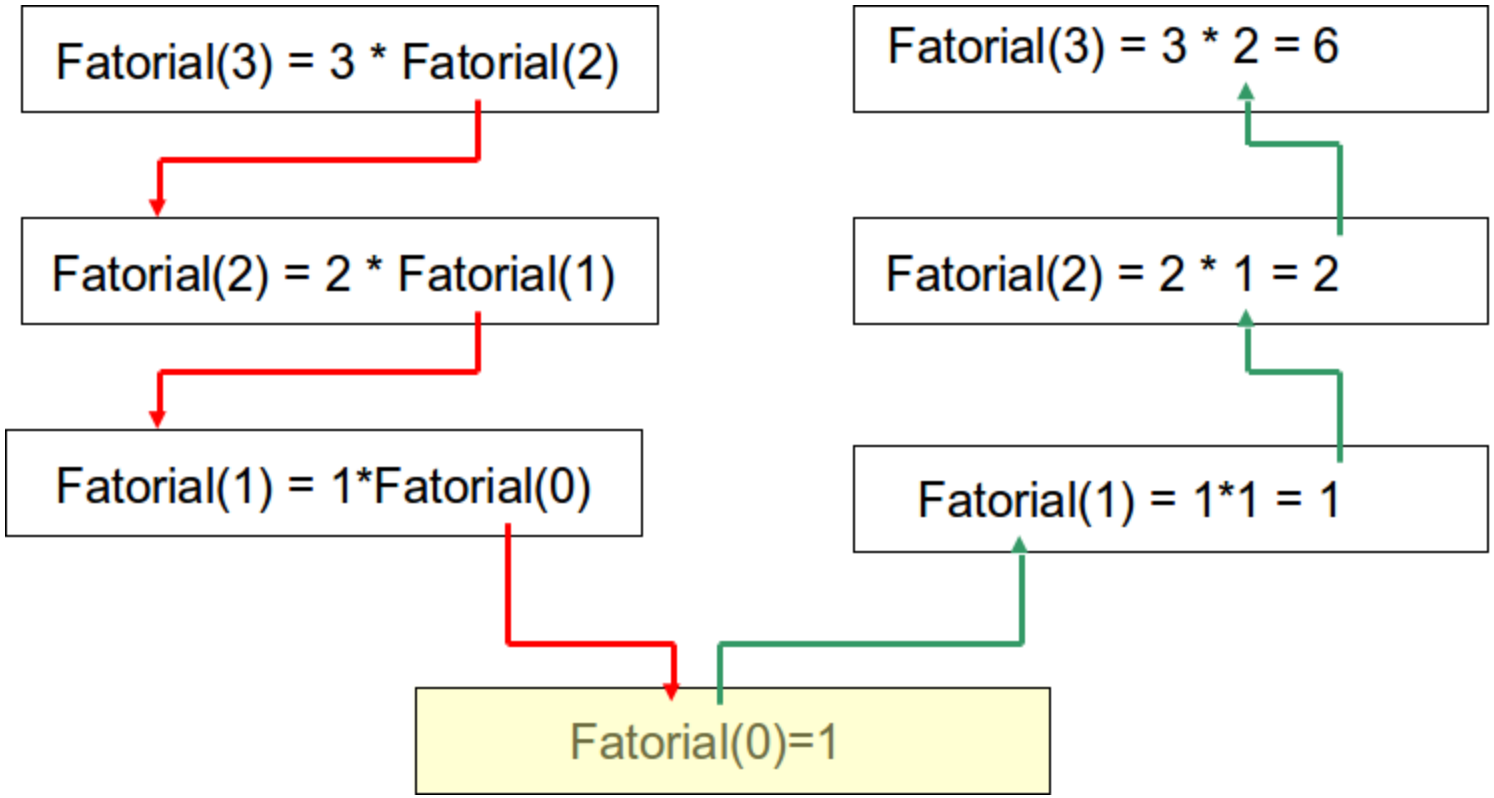
```
void Fat (int n) {
   if (n<=0)
      return 1;
   else
      return n * Fat(n-1);
}

Main() {
   int n = 5;
   int f = fat(n);
   cout << "Fatorial de " << n << " é " << f << endl;
}
```

Condição de Parada

Chamada base - Recursiva

Exemplo Fatorial



Exemplo Fatorial

Solução Iterativa

```
long Fatorial(int n){
    // declarações locais
    long factN = 1;
    int i;

    for (i = 1; i <= n; i++)
        factN = factN * i;

    return factN;
}
```

Exemplo Fibonacci

Série de Fibonacci:
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89...

Qual é a lógica?

- $F_n = F_{n-1} + F_{n-2} \qquad n > 2,$
- $F_0 = F_1 = 1$

```
int Fib(int n) {
    if (n<2)
        return 1;
    else
        return Fib(n-1) + Fib(n-2);
}
```

QUAL O RESULTADO?

```
#include <cmath>
#include <iostream>
using namespace std;
int funcao(int x)
{
  cout << "\n" << x;
  if(abs(x) < 10 )
    return 1;
  else
    return(1 + funcao(x/10));
 }
int main()
{
  int num;
  num=10145;
  cout <<" Total: "  << funcao(num) << endl;

  return 0;
}
```

PERCORRER UM VETOR RECURSIVAMENTE

- A função chama a si mesma recursivamente em uma versão menor da entrada (n - 1):

```
funcao procura(x, v, n) {
 se (v[n] == x)
    retorne n
 senao
    retorne procura (x,v,n-1)
}
```

EXERCÍCIOS

1. Implemente um algoritmo recursivo do somatório de 0 ate n.
2. Fazer um programa que leia,some 2 valores inteiros e mostre o resultado da soma. No final do programa, deverá ter uma recursividade que chame novamente o programa principal, mostre a mensagem “Digite 1 se desejar executar o programa novamente”, caso positivo, executar o programa novamente caso negativo, terminar a execução do programa.
3. Implemente a função recursiva **procura** do slide anterior.
4. Implemente uma função recursiva e outra iterativa de:
   a. Fibonacci
   b. Fatorial
   c. Potencia a^b
   d. MDC
5. Implemente uma função recursiva que inverte um vetor de tamanho n.
6. Implemente a solução da torre de Hanoi.
7. Implemente uma função recursiva que faça uma busca binária de um vetor n ordenado.

BONS ESTUDOS :)

Algoritmos

Prof. Demétrios Coutinho

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Campus Pau dos Ferros
Disciplina de Algoritmos